1.DATA LOADING AND PRE-PROCESSING

In [13]:
```
import pandas as pd
data=pd.read_csv("C:\\Users\\YOGA\\Downloads\\nlp_dataset.csv")
data
```

Out[13]:

	Comment	Emotion
0	i seriously hate one subject to death but now ...	fear
1	im so full of life i feel appalled	anger
2	i sit here to write i start to dig out my feel...	fear
3	ive been really angry with r and i feel like a...	joy
4	i feel suspicious if there is no one outside l...	fear
...	...	...
5932	i begun to feel distressed for you	fear
5933	i left feeling annoyed and angry thinking that...	anger
5934	i were to ever get married i d have everything...	joy
5935	i feel reluctant in applying there because i w...	fear
5936	i just wanted to apologize to you because i fe...	anger

5937 rows × 2 columns

In [14]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5937 entries, 0 to 5936
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   Comment  5937 non-null   object
 1   Emotion  5937 non-null   object
dtypes: object(2)
memory usage: 92.9+ KB
```

In [15]:
```
data.isnull().sum()
```

Out[15]:
```
Comment    0
Emotion    0
dtype: int64
```

In [16]:
```
data.describe()
```

Out[16]:

	Comment	Emotion
count	5937	5937
unique	5934	3
top	i feel like a tortured artist when i talk to her	anger
freq	2	2000

In [18]:
```
data.duplicated()
```

Out[18]:
```
0       False
1       False
2       False
3       False
4       False
        ...
5932    False
5933    False
5934    False
5935    False
5936    False
Length: 5937, dtype: bool
```

In [19]:
```
data.shape
```

Out[19]:
```
(5937, 2)
```

#TEXT CLEANING

In [24]:
```
import re
def clean_text(text):
    # Remove non-alphanumeric characters
    text = re.sub(r'\W', ' ', text)
    data['cleaned_text'] = data['Comment'].apply(clean_text)
print((data['cleaned_text']))
```

```
0       seriously hate one subject death feel reluctan...
1                              im full life feel appalled
2       sit write start dig feelings think afraid acce...
3       ive really angry r feel like idiot trusting fi...
4       feel suspicious one outside like rapture happe...
                              ...
5932                                 begun feel distressed
5933    left feeling annoyed angry thinking center stu...
5934    ever get married everything ready offer got to...
5935    feel reluctant applying want able find company...
5936           wanted apologize feel like heartless bitch
Name: cleaned_text, Length: 5937, dtype: object
```

#TOKENIZATION

In [21]:
```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\YOGA\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[21]: True

In [26]:
```
from nltk.tokenize import word_tokenize
data['tokens']=data['cleaned_text'].apply(nltk.word_tokenize)
data['tokens']
```

Out[26]:
```
0       [seriously, hate, one, subject, death, feel, r...
1                          [im, full, life, feel, appalled]
2       [sit, write, start, dig, feelings, think, afra...
3       [ive, really, angry, r, feel, like, idiot, tru...
4       [feel, suspicious, one, outside, like, rapture...
                              ...
5932                          [begun, feel, distressed]
5933    [left, feeling, annoyed, angry, thinking, cent...
5934    [ever, get, married, everything, ready, offer,...
5935    [feel, reluctant, applying, want, able, find, ...
5936    [wanted, apologize, feel, like, heartless, bitch]
Name: tokens, Length: 5937, dtype: object
```

#STOP WORD REMOVAL

In [27]:
```
from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\YOGA\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[27]: True

In [29]:
```
stop_words=set(stopwords.words('english'))
stop_words
```

Out[29]:
```
{'a',
 'about',
 'above',
 'after',
 'again',
 'against',
 'ain',
 'all',
 'am',
 'an',
 'and',
 'any',
 'are',
 'aren',
 "aren't",
 'as',
 'at',
 'be',
 'because',
```

2.FEATURE EXTRACTION

In [31]:
```
data
```

Out[31]:

	Comment	Emotion	cleaned_text	tokens
0	i seriously hate one subject to death but now ...	fear	seriously hate one subject death feel reluctan...	[seriously, hate, one, subject, death, feel, r...
1	im so full of life i feel appalled	anger	im full life feel appalled	[im, full, life, feel, appalled]
2	i sit here to write i start to dig out my feel...	fear	sit write start dig feelings think afraid acce...	[sit, write, start, dig, feelings, think, afra...
3	ive been really angry with r and i feel like a...	joy	ive really angry r feel like idiot trusting fi...	[ive, really, angry, r, feel, like, idiot, tru...
4	i feel suspicious if there is no one outside l...	fear	feel suspicious one outside like rapture happe...	[feel, suspicious, one, outside, like, rapture...
...	...	...	...	...
5932	i begun to feel distressed for you	fear	begun feel distressed	[begun, feel, distressed]

TfidfVectorizer

In [8]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(df['cleaned_text'])

print(X.shape)
```

```
(5937, 3)
```

3. MODEL DEVELOPMENT

A) NAIVE BAYES

In [30]:
```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report


y = data['Emotion']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, r


nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)

y_pred_nb = nb_model.predict(X_test)
print("Naive Bayes Classification Report:")
print(classification_report(y_test, y_pred_nb))
```

```
Naive Bayes Classification Report:
              precision    recall  f1-score   support

       anger       1.00      1.00      1.00       600
        fear       1.00      1.00      1.00       614
         joy       1.00      1.00      1.00       568

    accuracy                           1.00      1782
   macro avg       1.00      1.00      1.00      1782
weighted avg       1.00      1.00      1.00      1782
```

B) SUPPORT VECTOR MACHINE-[SVM]

In [11]:
```
from sklearn.svm import SVC
svm_model = SVC()
svm_model.fit(X_train, y_train)


y_pred_svm = svm_model.predict(X_test)
print("SVM Classification Report:")
print(classification_report(y_test, y_pred_svm))
```

```
SVM Classification Report:
              precision    recall  f1-score   support

       anger       1.00      1.00      1.00       600
        fear       1.00      1.00      1.00       614
         joy       1.00      1.00      1.00       568

    accuracy                           1.00      1782
   macro avg       1.00      1.00      1.00      1782
weighted avg       1.00      1.00      1.00      1782
```

4. MODEL COMPARISON

In [12]:
```
from sklearn.metrics import accuracy_score

# Naive Bayes
nb_accuracy = accuracy_score(y_test, y_pred_nb)
nb_f1 = classification_report(y_test, y_pred_nb, output_dict=True)['weight

# SVM
svm_accuracy = accuracy_score(y_test, y_pred_svm)
svm_f1 = classification_report(y_test, y_pred_svm, output_dict=True)['weig

print(f"Naive Bayes - Accuracy: {nb_accuracy}, F1-Score: {nb_f1}")
print(f"Support Vector Machine - Accuracy: {svm_accuracy}, F1-Score: {svm_
```

```
Naive Bayes - Accuracy: 1.0, F1-Score: 1.0
Support Vector Machine - Accuracy: 1.0, F1-Score: 1.0
```